Introducing Complexity Curtailing Techniques for the Tour Construction Heuristics for the Travelling Salesperson Problem

Hindawi Publishing Corporation
Journal of Optimization
Volume 2016, Article ID 4786268, 15 pages
http://dx.doi.org/10.1155/2016/4786268

Research Article

Introducing Complexity Curtailing Techniques for the Tour Construction Heuristics for the Travelling Salesperson Problem

Ziauddin Ursani[1,2] and David W. Corne[1,2]

[1]*School of Mathematical and Computer Sciences, Heriot Watt University, Edinburgh EH14 4AS, UK*
[2]*Route Monkey Ltd., Livingston EH54 5DW, UK*

Correspondence should be addressed to Ziauddin Ursani; ziaursani@yahoo.com

Received 22 March 2016; Revised 23 May 2016; Accepted 8 June 2016

Academic Editor: Ling Wang

In this paper, complexity curtailing techniques are introduced to create faster version of insertion heuristics, that is, cheapest insertion heuristic (CIH) and largest insertion heuristic (LIH), effectively reducing their complexities from $O(n^3)$ to $O(n^2)$ with no significant effect on quality of solution. This paper also examines relatively not very known heuristic concept of max difference and shows that it can be culminated into a full-fledged max difference insertion heuristic (MDIH) by defining its missing steps. Further to this the paper extends the complexity curtailing techniques to MDIH to create its faster version. The resultant heuristic, that is, fast max difference insertion heuristic (FMDIH), outperforms the "farthest insertion" heuristic (FIH) across a wide spectrum of popular datasets with statistical significance, even though both the heuristics have the same worst case complexity of $O(n^2)$. It should be noted that FIH is considered best among lowest order complexity heuristics. The complexity curtailing techniques presented here open up the new area of research for their possible extension to other heuristics.

1. Introduction

The Traveling Salesman Problem (TSP) is one of the most studied problems in the scientific literature and sometimes is referred to as a mother of all combinatorial optimization problems. It continues to be a testing ground for the development of combinatorial optimization methods, while having numerous practical applications in diverse areas, including logistics, genetics, manufacturing, telecommunications, and neuroscience [1]. Examples of real-world application domains with problems that can be naturally formulated as the TSP include VLSI design, vehicle routing, data clustering, and job-shop scheduling [2]. The earliest reference to the TSP can be found in the 1832 German handbook for travelling salesmen [1]. The problem consists of finding the shortest possible tour of a set of n cities, such that the tour starts from and ends at the same city and visits each of the remaining cities precisely once. The problem is simple to state but has proved to be intractable and is included among the seven "millennium prize problems" described by the Clay Mathematical Institute, carrying a prize of one million US dollars for discovery of a polynomial time solution method.

Meanwhile, research and practice in the TSP have focused on heuristic methods that yield fast approximate solutions. These heuristic methods tend to cluster into three main groups: (i) tour construction heuristics, (ii) tour improvement heuristics, and (iii) composite heuristics (which combine elements of both tour construction and tour improvement). A tour construction heuristic constructs the tour from scratch, beginning with one city and iteratively expanding the subtour by one city at a time. In contrast, a tour improvement heuristic begins with a complete tour and makes one or more rearrangements in an attempt to improve it. A vast space of "composite" heuristics obscures the distinction between these two categories by using elements of both; for example, in the highly successful Lin-Kernighan heuristic [3], the context is that of iterated tour improvement; however the improvement process consists of repeated construction of full solutions from partial solutions. The concord software code [4], which has solved most problems in TSPLIB to optimality,

uses the Lin Kernighan heuristic to find out near optimal tours and then applies various mathematical programming methodologies to achieve optimality.

With no general polynomial time solution method yet available, the quest for the development of new heuristics for the TSP remains active. Tour construction heuristics have played an important role in this quest, in particular because they can form components of a wide range of approaches. For example, they can be used to construct the starting solutions for tour improvement heuristics [5], and they can provide rough estimates of the cost of optimal solutions. In turn, they therefore provide interesting analytical grounds for the study of upper bounds on solution quality [6] and provide material for empirical studies [7, 8] that attempt to understand how optimal solutions differ from solutions that arise from tour construction heuristics. Such studies lead to better understanding of the structure of optimal solutions of the TSP, a problem that has now remained centre of our intellectual curiosity for centuries.

The basic procedure of tour construction heuristics can be summarised as follows:

(1) Establishment of the initial small subtour (subtour establishment rule).

(2) Selection of a city not present in the current subtour (selection rule).

(3) Expansion of the current subtour to include the selected city (expansion rule).

(4) Iterative application of steps (2) and (3) until a complete tour is obtained.

In the repeated application of steps (2) and (3), the subtour is "successively augmented" with the insertion of a new city; this is why tour construction heuristics are sometimes also called "successive augmentation" heuristics (see, e.g., [9]). Different tour construction heuristics are characterised by the specific methods they choose for each of the three steps: the initial subtour establishment, selection, and expansion rules. Expansion rules can be grouped into two main types: insertion and addition. An insertion-based expansion rule chooses where in the permutation to place the new city on the basis of the cost of the resulting subtour, whereas an addition-based expansion rule bases this decision on next-hop distance. It has been reported that insertion heuristics generally perform better than addition heuristics (see, e.g., [8]); we have therefore chosen to focus on insertion-based expansion heuristics in this paper.

Three insertion heuristics are particularly prominent in the literature, namely, nearest, farthest, and cheapest [10]. We consider these three heuristics in the remainder of this work, along with a further two that are less well known. The first of these two is the "largest-cost insertion heuristic" (LIH) [11], which sits naturally alongside the aforementioned three, despite being rarely considered in the literature. The second is the "max difference insertion heuristic" (MDIH). Tunnel and Heath [12] presented "max difference" as a concept that could be attached to any other heuristic; however we argue that, with appropriate extensions that we later describe (resulting in MDIH), it is more appropriately seen as a tour construction heuristic in itself, and in fact we demonstrate that it is a particularly effective one.

The insertion heuristics under study can be divided into two groups, that is, distance based insertion heuristics (DBIH) and cost based insertion heuristics (CBIH). The DBIH has city selection rule based on distance. This group includes nearest (NIH) and farthest insertion heuristics (FIH). The CBIH has city selection rule based on the cost of insertion (see (5)). This group includes cheapest (CIH), largest (LIH), and max difference insertion heuristics (MDIH). The difference in selection rule brings about difference in worst case complexity of heuristics with DBIH having worst case complexity of $O(n^2)$ and CBIH having complexity of $O(n^3)$ (see Section 4). This paper proposes techniques to curtail the complexity of CBIH to $O(n^2)$, effectively creating faster versions of CBIH, that is, FCIH, FLIH, and FMDIH, while addition of "F" denotes the word fast.

Previously it has been reported that farthest insertion heuristic (FIH) generally performs best among the community of insertion heuristics of $O(n^2)$ or lower time complexity (see, e.g., [13]). However we will show that the performance of FMDIH is consistently better than FIH on a wide spectrum of popular datasets even though the complexity of FMDIH is no greater than that of FIH.

The remainder of this paper is structured as follows. In Section 2, we describe four of the five insertion heuristics of interest: cheapest, largest, nearest, and farthest insertion. In Section 3 we describe the previously overlooked "max difference" concept and build on that to describe the "max difference insertion" heuristic. In Section 4, we discuss design and complexity issues for all five of the heuristics of interest, and in Section 5 we then describe our complexity curtailing techniques to produce accelerated and approximated variants of cost based insertion heuristics. Section 6 provides summaries of empirical results, and we conclude with a statement of our main findings in Section 7. In Appendix, we then show the empirical results for the farthest and max difference insertion heuristics in finer detail.

2. Description of Tour Construction Heuristics

Below, we explain the design details of four of the five main heuristics considered in this paper. Our elaboration of the details follows the four-part structure given in Section 1 for tour construction heuristics. Being insertion heuristics, the most important elements are the city selection and subtour expansion methods. The city selection method is also salient for another reason, which is the fact that the name of a tour construction heuristic (e.g., cheapest and farthest) tends to reflect the nature of this rule. We come back to this fact in Section 3, when we speculate about why the "max difference" heuristic has been largely overlooked. The simple "initial subtour establishment" rule is also described for completeness and to facilitate replication of our experiments. Meanwhile, it is worth noting an interesting aspect of the city selection rule.

2.1. Initial Subtour Establishment. This is the first step in any tour construction heuristic. In each of the standard

forms of the cheapest, largest, nearest, and farthest insertion heuristics, this initial "subtour" is simply a single city chosen uniformly at random.

2.2. City Selection. Momentarily referring to the subtour expansion rule (in Section 2.3), we note that two of the heuristics of interest are based on cost, and two are based on distance. Where subtour expansion is based on cost, the city selection rule is as follows. Let τ be the set of cities in the subtour and let τ' be the set of cities not in the subtour, while city $k \in \tau'$ and $\text{cost}(\tau, k)$ is the expansion cost of subtour τ by insertion of city k. Then, the city k is selected such that

for CIH

$$\text{cost}(\tau, k) = \min\left\{\forall j \in \tau' \ \left(\text{cost}(\tau, j)\right)\right\} \quad (1)$$

and for LIH

$$\text{cost}(\tau, k) = \max\left\{\forall j \in \tau' \ \left(\text{cost}(\tau, j)\right)\right\}. \quad (2)$$

When subtour expansion is instead based on distance, the city selection rule is different. Now, let $d(h, k)$ be the distance between cities k and h; the city k is selected such that

for NIH

$$d(h, k) = \min\left\{\forall j \in \tau', \ \forall i \in \tau \ \left(d(i, j)\right)\right\} \quad (3)$$

and for FIH

$$d(h, k) = \max\left\{\forall j \in \tau', \ \forall i \in \tau \ \left(d(i, j)\right)\right\}. \quad (4)$$

2.3. Subtour Expansion. As indicated in the introduction, subtour expansion heuristics can be based around either *insertion* or *addition*. We are only concerned with insertion-based variants in this paper and so will confine our description accordingly. Let $k \in \tau'$ and $(i, i') \in \tau$, such that i and i' are a pair of consecutive cities in τ. Let $d(i, k)$ be the distance between the cities i and k. The subtour is expanded by inserting the selected city k in the subtour τ such that

$$\text{cost}(\tau, k) = \min\left[\forall (i, i') \in \tau \ \left\{d(i, k) + d(i', k) - d(i, i')\right\}\right]. \quad (5)$$

This step is applicable to each of CIH, LIH, FIH, and NIH. From this point onwards, we refer to the edge (i, i') related to $\text{cost}(\tau, k)$ as the *expansion cost edge*. It should be noted that the "largest-cost insertion" heuristic is not published yet, however, an unpublished paper about it is present on the internet [10]. We include it in our experiments, noting that it is the logical counterpoint to CIH, in the same way that FIH is related to NIH. It therefore completes a set of four heuristics of which two are based on minimum and maximum distance ((3)-(4)) while two are based on minimum and maximum expansion cost ((1)-(2)). We later find that LIH outperforms both CIH and NIH, though apparently missing from the literature.

3. The Max Difference Insertion Heuristic (MDIH)

The idea behind the MDIH was introduced in a Master's thesis [12] a quarter of a century ago and no research paper about this could be found in literature. The "max difference" concept was presented by Tunnel and Heath as a way to engineer new variants of existing heuristics (they applied it to two versions of CIH and also to Stewart's Algorithm [14]). However, since (as we will see) the concept relates directly to how the city selection step is done, from which the names of tour construction heuristics seem invariably to be derived, we would argue that it merits reinvention as a fully fledged tour construction heuristic. Its performance in the form of MDIH, as detailed in the next section, certainly supports this view. Meanwhile we speculate that Tunnel and Heath may not have put it forward as a standalone new heuristic, based on the belief that it was the "insertion" (subtour expansion) rather than the "max difference" (city selection) that was salient in such categorization; in turn, this seems to have led to its being unnoticed, despite its comparatively strong performance among similar heuristics.

We now explain the key "city selection" step for MDIH. First, we need some helper definitions. The first of these is the function "nth minimum," denoted as $\min_n\{\forall v \in U \ (v)\}$. The meaning of "$n$th minimum" is that if all the values v in set U are sorted from minimum to maximum then this function represents the nth value in that list. Now, the "nth expansion cost" of a subtour by the insertion of any city not in that subtour is equal to the function $\min_n\{\forall v \in U \ (v)\}$, if the set U is the set of all the expansion costs of that subtour on all of its edges caused by the insertion of that city. Now let $k \in \tau'$ and $(i, i') \in \tau$, such that i and i' are a pair of consecutive cities in τ. Let $d(i, k)$ be the distance between the cities i and k. Let $\text{cost}_n(\tau, k)$ represent nth expansion cost of the tour τ by the insertion of city k and then by the definition

$$\text{cost}_n(\tau, k) = \min_n\left[\forall (i, i') \in \tau \ \left\{d(i, k) + d(i', k) - d(i, i')\right\}\right]. \quad (6)$$

It should be noted that, apart from here, the exclusive use of above function is made in Sections 5.2 and 5.3. From this point onwards the edge (i, i') related to $\text{cost}_n(\tau, k)$ is referred to as the nth expansion cost edge. In the city selection rule of MDIH the city k is selected such that the cost difference $D(k)$ between its expansion cost and 2nd expansion cost is maximum among all the cities not in the tour; that is,

$$D(k) = \max\left[\forall k \in \tau' \ \left\{\text{cost}_2(\tau, k) - \text{cost}(\tau, k)\right\}\right]. \quad (7)$$

It should be noted that there is no difference between $\text{cost}_1(\tau, k)$ and $\text{cost}(\tau, k)$. The 3rd step (subtour expansion) follows the same rule as represented by (5).

Finally to present MDIH as complete tour construction heuristic in itself, we also explore five variants of the subtour establishment rule (step (1)). Unlike the other insertion heuristics considered in this paper, the city selection step of MDIH requires an initial subtour containing at least three

cities. We therefore test five subtours establishment rules: (i) a subtour of three randomly chosen cities; (ii) a subtour first formed by two randomly chosen cities and then the third city is chosen based on cheapest expansion; (iii) a subtour first formed by two randomly chosen cities and then the third city is chosen based on largest-cost expansion; (iv) a subtour initially formed by one randomly chosen city and the next two are chosen iteratively based on cheapest expansion; (v) a subtour first formed by one randomly chosen city and then the next two are iteratively chosen on the basis of largest-cost expansion. Associated experiments and results are discussed later in Section 6.

4. Design and Complexity Aspects of Tour Construction Heuristics

Design of any heuristic consists of the design of data structures and algorithms for its efficient implementation. Our implementation centres around two key data structures: a doubly circular linked list representing the set of cities not in the subtour and a singly circular linked list representing the subtour. For set of cities not in the subtour, the doubly circular linked list allows for a very simple and fast city deletion operation. For the set of cities representing the subtour a, single circular linked list is sufficient since only an insertion operation is needed here. Initially, a doubly circular linked list is prepared representing the set of cities not in the subtour. From this list, cities are deleted one by one and correspondingly inserted in the single circular linked list representing the subtour of cities at each step. The procedure continues until the doubly circular linked list becomes empty.

The design of step (2) (city selection) of DBIH, that is, the nearest and farthest insertion heuristics ((3)-(4)), consists of finding the nearest neighbour city present in the subtour for each of the cities not in the subtour. To reduce computation costs, the nearest neighbour city in the subtour for each city not in the subtour is recorded in each iteration, and in the next iteration the distance of only the recently added city is compared with the distance of the nearest neighbour from the previous iteration, thus updating the nearest neighbour information for the current iteration. Therefore, FIH and NIH can be implemented within time complexity of $O(n^2)$.

In the case of CBIH, that is, the cheapest and largest insertion heuristics, the design of the city selection step ((1) and (2)) consists of finding the expansion cost for insertion into the subtour of each city not in the subtour. This implies that the expansion cost needs to be computed for each city not in the subtour; in turn, computation of the expansion requires visits to each and every edge of the subtour. To make this efficient, information should be preserved between iterations, and in each new iteration the expansion cost of only newly added edges is computed and compared with the expansion costs of the previous iteration to update the expansion cost information. However this shortcut may not be possible for all the cities not in the subtour. This is because some of the cities may have expansion costs in the previous iteration at an edge which is broken in the current iteration. This means the information about the expansion cost of those cities is no longer valid since the relevant edge for the expansion does not exist. For such cities, we need to compute the expansion costs for newly formed edges and if this expansion cost is less than or equal to their expansion cost in previous iterations, then it means their new expansion cost has been found on the newly formed edges. However, if not, we are forced to recompute the expansion cost of these cities on all edges of the current subtour to find out their new expansion cost. Therefore worst case scenario complexity of CIH and LIH is $O(n^3)$. The average complexity of CIH is reported as of the order $O(n^2 \log n)$ [14].

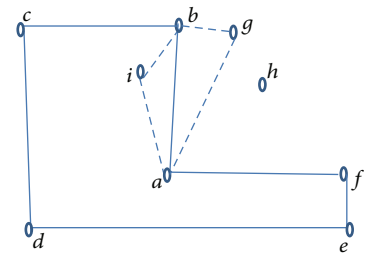

Figure 1: Subtour "*abcdefa*" showing principle of fast CIH.

To design the city selection step for MDIH, for each city not in the subtour both the expansion cost and the 2nd expansion cost need to be computed (7). However, both of these quantities can be computed simultaneously in a single scan of the subtour. To implement the step efficiently, these quantities should be preserved between iterations for each city not in the subtour, and in the new iteration only the expansion cost and 2nd expansion cost of the recently added edges are computed and compared with those of the previous iteration to update them. However, just as was the case for CIH and LIH, this shortcut is not always possible; for some cities the expansion costs at all edges of the subtour need to be computed. Therefore the worst case scenario complexity of MDIH is same as that of cheapest and largest insertion heuristics, that is, $O(n^3)$.

5. Complexity Curtailing Techniques for Cost Based Insertion Heuristics

In this section, complexity curtailing techniques are introduced to create fast and approximate variants of CIH, LIH, and MDIH.

5.1. Design of Fast Cheapest Insertion Heuristic (FCIH). We can devise a new but faster heuristic based on CIH on the basis of simple geometrical principle. Consider the subtour "*abcdefa*" in Figure 1.

In Figure 1 it can be seen that g, h, i are the cities yet to be inserted. For each of these cities, the cheapest insertion edge is ab. The final shape of the tour depends on the order in which these three edges are inserted. If city i is inserted first, then edge ab will be deleted and the edges ia and ib will be added. In the new subtour $aibcdefa$ it is now a genuine possibility that the cheapest insertion for city h becomes edge af, since edge ab is no longer present. In this case, we will need to traverse the entire current subtour $aibcdefa$ to determine the

Procedure FCIH starts
Form circular linked list of cities τ' not in the subtour
Form empty Linked list τ
Choose an arbitrary city $p \in \tau'$
Delete p from the list τ'
Insert p in the empty list τ
Initialize expansion costs of subtour τ by insertion w.r.t. all h in τ' to very high value
while τ' is not empty *do begin*
    do begin (Visit all h in τ')
        Let
        $\delta[1]$ = Cost of insertion of city h on 1st newly formed edge
        $\delta[2]$ = Cost of insertion of city h on 2nd newly formed edge
        $\delta[3] = \text{cost}(\tau, h)$
        If expansion cost edge is broken in last iteration
        then $\text{cost}(\tau, h) = \min[\forall r \in \{1, 2\}\ (\delta[r])]$
        else $\text{cost}(\tau, h) = \min[\forall r \in \{1, 2, 3\}\ (\delta[r])]$
        end if
    while all h in τ' not visited
    Choose city q in τ' such that $\text{cost}(\tau, q) = \min\{\forall h \in T\ (\text{cost}(\tau, h))\}$
    Delete q from the list τ'
    Insert q in τ on the edge connected to $\text{cost}(\tau, q)$
end while
Procedure FCIH ends

Procedure 1: Fast cheapest insertion heuristic (FCIH).

new cheapest insertion for h. However, if city g had been inserted first instead of city i then edge ab is deleted and new edges ga and gb are added. In this case most likely situation becomes that h is inserted in the newly added edge ga, requiring only inspection of the newly added edges ga and gb.

In interim summary, if we encounter the scenario in Figure 1 during the construction of a tour using CIH, then, for the sake of computational efficiency, we would hope that city i is not chosen as the first of the three to be inserted. However, some reflection will reveal that the position of city i in this scenario is unlikely in the first place. It is well known that CIH grows tours "outwardly." This means that the cities not in the subtour lie outside the periphery of the subtour, rather than inside it, such as we see with city i in Figure 1. By exploiting this property, we can propose a variant of CIH that only inspects the recently added edges in each iteration, even for the cities whose earlier cheapest edge is broken. Though no longer guaranteeing the "cheapest" insertion, intuition suggests that this would lead to a favourable tradeoff, with at most minor loss of quality set against significant benefit in speed. In this "fast CIH" heuristic, the total number of computations of expansion costs (now independent of details of the dataset) is given below:

$$N_c = 2 \times \{(n-1) + (n-2) + (n-3) + \cdots + 3 + 2 + 1\} = n(n-1). \tag{8}$$

Meanwhile, since one expansion cost involves traversal of 3 edges, the total number of edges traversed for computation of expansion costs is given by

$$N_e = 3n(n-1). \tag{9}$$

The time complexity of "fast CIH" is therefore $O(n^2)$, which is below the average complexity of CIH. Note that fast CIH runs in the order $O(n^2)$ on any dataset (with best case, worst case, and average case scenarios all equivalent). A full procedural summary of this new "fast cheapest insertion" heuristic (FCIH) is given in Procedure 1.

The first five lines of "Procedure 1" establish the initial tour. Expansion of this initial tour is then implemented in the remainder of the procedure. The inner loop in the remainder is used to compute the expansion cost $\text{cost}(\tau, h)$ for each city h (the definition of this cost is given in (5)). However, a short cut is used to compute this cost, by computing the expansion cost on only two newly formed edges. There are two potential scenarios associated with this shortcut: in the first scenario, the edge related to the expansion cost of this city in the previous iteration is broken, and in the second scenario it has survived. In the first scenario, the new expansion cost of the city h in the current iteration is assumed to be the minimum of the two expansion costs of city h computed on the two newly formed edges; in the second scenario, the difference is that we *also* consider the expansion costs of city h computed in the previous iteration. The condition "*while* all h in τ' not visited" ensures that the cost of all uninserted cities is computed. The outer loop chooses the city for insertion that has the lowest expansion cost.

5.2. Design of Fast Largest Insertion Heuristic (FLIH). In contrast to CIH, tours constructed by LIH do not grow "outwardly," and the situation in Figure 1, where one of the currently unvisited cities lies inside the periphery of the current subtour, is a common occurrence. We cannot therefore design a faster version of LIH on the same basis used

Procedure FLIH starts
Form circular linked list of cities τ' not in the subtour
Form empty Linked list τ
Choose an arbitrary city $p \in \tau'$
Delete p from the list τ'
Insert p in the empty list τ
Initialize expansion costs and 2nd expansion costs w.r.t. all h in τ' to very high value
while τ' is not empty do begin
    do begin (Visit all h in τ')
        Let
        $\delta[1]$ = Cost of insertion of city h on 1st newly formed edge
        $\delta[2]$ = Cost of insertion of city h on 2nd newly formed edge
        $\delta[3] = \text{cost}(\tau, h)$
        $\delta[4] = \text{cost}_2(\tau, h)$
        If expansion cost edge is broken in last iteration
        then $\forall n \in \{1, 2\}\ (\text{cost}_n(\tau, h) = \min_n[\forall r \in \{1, 2, 4\}\ (\delta[r])])$
        else if 2nd expansion cost edge is broken in last iteration
        then $\forall n \in \{1, 2\}\ (\text{cost}_n(\tau, h) = \min_n[\forall r \in \{1, 2, 3\}\ (\delta[r])])$
        else $\forall n \in \{1, 2\}\ (\text{cost}_n(\tau, h) = \min_n[\forall r \in \{1, 2, 3, 4\}\ (\delta[r])])$
        end if
    while all h in τ' not visited
    Choose city q in τ' such that $\text{cost}(\tau, q) = \max\{\forall h \in T\ (\text{cost}(\tau, h))\}$
    Delete q from the list τ'
    Insert q in τ on the edge connected to $\text{cost}(\tau, q)$
end while
Procedure FLIH ends

Procedure 2: Fast largest insertion heuristic (FLIH).

for FCIH. However, in the case of LIH a different approximate assumption can be made which may help avoid computing expansion costs on all edges of the current subtour. Referring again to Figure 1: if city i is inserted in the subtour first, the new subtour *aibcdefa* is formed, breaking the edge *ab* as a result. Now in this case, as discussed earlier, city h might not make its cheapest insertion with newly formed edges *ai* and *ib*. However, there is a genuine possibility that city h makes its 2nd cheapest insertion with *ai* or *ib*. This leads us to the idea of a fast version of LIH in which we approximate, rather than guarantee, the cheapest insertion, while scanning at most four edges in each iteration of the construction process. In detail, in each iteration we only need to keep a record of the 2nd expansion cost in addition to the (1st) expansion cost for each city not in the subtour. In turn, we find the 2nd expansion cost of a city by scanning at most only four edges: the two newly formed edges, along with the expansion cost edge and 2nd expansion cost edge of the previous iteration. If either of the latter two were broken in the previous iteration, then we can simply use the remaining three edges to (approximately) update the expansion cost and 2nd expansion cost in the new iteration. Again, the resulting heuristic, FLIH, has complexity $O(n^2)$, independently of details of the dataset. A procedural summary of fast largest insertion heuristic (FLIH) is given in Procedure 2.

In common with Procedure 1, the first five lines of "Procedure 2" establish the initial subtour, while the remainder concern its expansion. In the inner loop, the expansion cost $\text{cost}(\tau, h)$ and second expansion cost $\text{cost}_2(\tau, h)$ are calculated for each city h (see (5) and (6), resp.). The costs are calculated on the basis of two newly formed edges, and the detail of the procedure concerns three potential scenarios that may arise. In the first scenario, the edge related to the expansion cost of city h in the previous iteration is broken. In the second, the edge related to the *second expansion cost* of city h is broken; in the third scenario, none of the expansion edges is broken. In the first scenario the new expansion cost and second expansion cost of city h are taken to be the two minimum expansion costs of three quantities: the two expansion costs of city h computed on the two newly formed edges and its second expansion cost in the last iteration. In the second scenario, the three quantities are now the two expansions costs computed on the two newly formed edges (as before) and its expansion cost computed in the previous iteration. Finally, in the third scenario, the two costs are taken as the smallest two of four quantities: the two expansion costs of city h computed on the two newly formed edges and its expansion cost and second expansion cost computed in the earlier iteration. The condition "*while* all h in τ' not visited" ensures that cost of all uninserted cities is computed. The outer loop chooses the city for insertion that has largest expansion cost.

5.3. Design of Fast Max Difference Insertion Heuristic (FMDIH). Faster design of MDIH is possible on a similar basis to that used in faster design of FLIH. A key difference between LIH and FLIH is that, in each iteration, LIH calculates only the expansion costs for the unvisited cities,

Procedure FMDIH starts
Form circular linked list of cities τ' not in the subtour
Form empty Linked list τ
Setup initial subtour of three cities $(p_1, p_2, p_3) \in \tau'$
Delete (p_1, p_2, p_3) from the list τ'
Insert (p_1, p_2, p_3) in the empty list τ
Initialize expansion costs, 2nd expansion costs and 3rd expansion costs w.r.t. all h in τ' to very high value
while τ' is not empty do begin
    do begin (Visit all h in τ')
        Let
        $\delta[1]$ = Cost of insertion of city h on 1st newly formed edge
        $\delta[2]$ = Cost of insertion of city h on 2nd newly formed edge
        $\delta[3] = \text{cost}(\tau, h)$
        $\delta[4] = \text{cost}_2(\tau, h)$
        $\delta[5] = \text{cost}_3(\tau, h)$
        If expansion cost edge is broken in last iteration
        then $\forall n \in \{1, 2, 3\}\ (\text{cost}_n(\tau, h) = \min_n[\forall r \in \{1, 2, 4, 5\}\ (\delta[r])])$
        else if 2nd expansion cost edge is broken in last iteration
        then $\forall n \in \{1, 2, 3\}\ (\text{cost}_n(\tau, h) = \min_n[\forall r \in \{1, 2, 3, 5\}\ (\delta[r])])$
        else if 3rd expansion cost edge is broken in last iteration
        then $\forall n \in \{1, 2, 3\}\ (\text{cost}_n(\tau, h) = \min_n[\forall r \in \{1, 2, 3, 4\}\ (\delta[r])])$
        else $\forall n \in \{1, 2, 3\}\ (\text{cost}_n(\tau, h) = \min_n[\forall r \in \{1, 2, 3, 4, 5\}\ (\delta[r])])$
        end if
    while all h in τ' not visited
    Choose city q in τ' such that cost difference $D(q) = \max[\forall h \in T\{\text{cost}(\tau, h) - \text{cost}_2(\tau, h)\}]$
    Delete q from the list τ'
    Insert q in τ on the edge connected to $\text{cost}(\tau, q)$
end while
Procedure FMDIH ends

Procedure 3: Fast max difference insertion heuristic (FMDIH).

while FLIH (approximately) computes both the expansion costs and the 2nd expansion costs. By going "one step further" in this sense, FLIH facilitates much faster update of this information, while sacrificing some exactness. To extend the same principle to fast MDIH, we will go "one step further" in the computations required for the city selection rule in MDIH. In MDIH, city selection requires both the expansion cost and the 2nd expansion cost for each city not in the subtour (7). Therefore, we can design a fast MDIH that additionally computes and records the 3rd expansion cost of each city not in the subtour. This means we need to record at most five edges per iteration: the two newly added, plus the expansion, 2nd expansion, and 3rd expansion cost edges from the previous iteration. As with FLIH, if one of the latter three was broken in the current iteration, we will simply rely on the others to provide approximations. A detailed procedural summary of the fast max difference insertion heuristic (FMDIH) is given in Procedure 3.

In "Procedure 3", again, the initial subtour is computed by the first five lines, and the remainder expands it. The inner loop of the remainder is used to compute the expansion cost $\text{cost}(\tau, h)$, the second expansion cost $\text{cost}_2(\tau, h)$, and the third expansion cost $\text{cost}_3(\tau, h)$ for each city h. The definitions of these costs are given in (5) and (6). The details of the inner loop achieve a shortcut in computing these costs, which involves four scenarios. In the first scenario, the edge related to the expansion cost of city h in the previous iteration is broken. In the second scenario, the edge related to the second expansion cost of city h is broken; in the third scenario, the edge related to the third expansion cost of city h is broken, and in the fourth scenario none of the three expansion edges is broken. In the first scenario the expansion cost, second expansion cost, and third expansion cost are assumed to be the smallest three of four quantities: the two expansion costs of city h computed on the two newly formed edges and its second and third expansion costs in the last iteration. In the second scenario they are taken as the smallest three of a slightly changed set of four quantities: the two expansion costs of city h computed on the two newly formed edges and its expansion cost and third expansion cost computed in the earlier iteration. In the third scenario, the four quantities are now the two expansion costs of city h computed on the two newly formed edges and its expansion cost and second expansion cost from the earlier iteration. In the fourth scenario, first, second, and third expansion costs are taken as minimum costs of five quantities, that is, two expansion costs on two newly formed edges and first, second, and third expansion costs of the previous iteration. The condition "*while* all h in τ' not visited" ensures that cost of all uninserted cities is computed. The outer loop chooses the city for insertion that has maximum cost difference (7).

6. Experiments and Results

This section consists of four subsections. In Section 6.1 we report experiments that evaluate each of the five types of initial subtour establishment rules described in Section 3 allowing us to "configure" and complete the new MDIH. Later in Section 6.2 performance of MDIH is compared with all baseline heuristics considered in this paper, that is, NIH, FIH, CIH, and LIH. In Section 6.3 the cost based insertion heuristics, that is, CIH, LIH, and MDIH, are compared with their faster versions FCIH, FLIH, and FMDIH. Finally in Section 6.4 the focus of our evaluation is to investigate the potential for FMDIH as a new $O(n^2)$ heuristic for fast approximate solution of TSPs, by comparing it with the current best-regarded $O(n^2)$ heuristic, that is, FIH.

Experiments are conducted on 109 datasets, involving from 14 to 15112 cities from a popular test bed [14]. For each heuristic and problem instance, 30 simulations are run independently, and we record the best, worst, and average solutions, the standard deviation in solution quality, and the average execution time. For the majority of experiments we provide summary statistics in this section (volume of results data precludes otherwise); however a full description of results is provided in an appendix for the FMDIH and FIH tests. The hardware used in the experiments was an Intel® Core™ i3-2348 M CPU @ 2.30 GHz, 8.0 GB Memory, with 64 bit operating system. The heuristics were coded in Microsoft Visual Studio C/C++.

6.1. Experiments to Finalize Initial Tour Establishment Step of Max Difference Insertion Heuristic (MDIH). First of all, in Table 1 we can see a summary of the results for MDIH with the five different strategies for generating the initial subtour. The heuristics named MDIH-1 to MDIH-5 represent MDIH using methods (i) to (v) as described in Section 3. Each column in Table 1 refers to a certain statistical value in relation to deviation from the known optimal solution. The mathematical description of each of those values is given below, where C_1 to C_5 in (10)–(14), respectively, represent the quantities in columns 1–5 in Table 1:

$$C_1 = \text{avg}_{j=1}^{j=N} \left\{ \min \left(\forall i \in S \ \left(\gamma_i^j \right) \right) \right\}, \tag{10}$$

$$C_2 = \text{avg}_{j=1}^{j=N} \left\{ \max \left(\forall i \in S \ \left(\gamma_i^j \right) \right) \right\}, \tag{11}$$

$$C_3 = \text{avg}_{j=1}^{j=N} \left\{ \text{avg}_{i=1}^{i=S} \left(\gamma_i^j \right) \right\}, \tag{12}$$

$$C_4 = \text{avg}_{j=1}^{j=N} \left[\text{avg}_{k=1}^{k=S} \left\{ \gamma_k^j - \text{avg}_{i=1}^{i=S} \left(\gamma_i^j \right) \right\} \right], \tag{13}$$

$$C_5 = \text{avg}_{j=1}^{j=N} \left\{ \text{avg}_{i=1}^{i=S} \left(t_i^j \right) \right\}, \tag{14}$$

where

$\gamma_i^j = ((v_i^j - v_0^j)/v_0^j) \times 100$,

$\text{avg}_{i=1}^{i=S}(\gamma_i^j) = (\sum_{i=1}^{i=S} \gamma_i^j)/S$,

N means number of datasets which equals 109,

S means number of simulations which equals 30,

v_i^j means cost of solution of a simulation i on a dataset j,

v_0^j means cost of optimal solution of a dataset j.

In Table 1, it can be seen that MDIH-5 has the best average solution quality. Its average quality for worst solution is also among the best, while its standard deviation is also the best of the five. Only on one criterion, average of best solution, its performance is worst among all. However, the difference between the best and worst values for this statistic is only 0.17%, and we propose that it can be ignored in favour of choosing method (v) as the appropriate initial subtour establishment rule for MDIH and FMDIH.

6.2. Experiments to Compare Performance of All Baseline Heuristics NIH, CIH, LIH, FIH, and MDIH. Table 2 summarises the results of the five tour construction heuristics NIH, CIH, LIH, FIH, and MDIH, with MDIH configured to use the fifth initial tour establishment rule. (The results for MDIH are already in Table 1 but are copied into Table 2 for convenient comparison.) From the results it can be seen that performance of MDIH is best among all heuristics on all five criteria. For example, the average deviation from the optimal solution for MDIH is 4.64%, and there is a relatively sharp decline until we come to the second best on this category, which is FIH at 7.27%. It seems that the performance of FIH is second best (to MDIH) in all criteria except standard deviation. Meanwhile, LIH seems to show the worst performance among those examined, on all criteria except the average deviation from the best solution. Closer inspection of the full results, however, shows that the relatively poor figures for LIH derive from a notable failure on only one problem instance, BRG180. On this instance its best solution is 260% (average + 22 standard deviations) away from the optimal, and its worst solution is 2032.82% (average + 59 standard deviations) away from the optimal. This dataset brings out the worst performance from each of the heuristics and is clearly a particularly unusual example with a structure that deviates from the vast majority of TSPs. In fact this problem arises from a participant-comparison metric in a cards-playing tournament, and performance of any heuristic on it is therefore not reflective of performance on cases where the matrix contents are more closely associated with physical distances. We therefore decided that omitting this dataset would lead to a more useful cross-comparison of the heuristics, and the outcome of that for the remaining 108 datasets is given in Table 3; MDIH is preserved as the best performer in all categories, while LIH is now third best.

In Figure 2 we attempt to visualise the performance of these five heuristics as a function of the number of cities, with number of cities on the horizontal axis, the vertical axis being percent difference of average solution of 30 simulations from the optimal solution. Since there is dense accumulation of datasets up to 2000 cities, the picture is not very clear up to this point. However after this point the pattern is quite clear. It can be seen that heuristics follow two different patterns of rise and falls of graph. One pattern is followed by MDIH, FIH, and LIH, while CIH and NIH follow another pattern. The MDIH-FIH-LIH pattern is dominated by MDIH, followed by FIH.

Table 1: Results of MDIH with five different initial subtour establishment rules.

Name	Average of best solution % age	Average of worst solution % age	Average of average solution % age	Average of standard deviation % age	Average of execution time seconds
(0)	(1)	(2)	(3)	(4)	(5)
MDIH-1	4.56	12.12	7.37	1.78	0.45
MDIH-2	4.53	11.28	6.75	1.50	0.45
MDIH-3	4.47	9.09	6.50	1.12	0.46
MDIH-4	4.56	9.65	6.71	1.29	0.48
MDIH-5	4.64	8.59	6.34	0.99	0.46

Table 2: Results of 5 different tour construction heuristics.

Name	Average of best solution % age	Average of worst solution % age	Average of average solution % age	Average of standard deviation % age	Average of execution time seconds
(0)	(1)	(2)	(3)	(4)	(5)
NIH	17.91	25.74	21.17	2.55	0.62
CIH	14.27	18.90	16.64	1.27	0.48
LIH	11.18	34.48	23.69	5.93	0.42
FIH	7.27	14.96	10.16	1.99	0.63
MDIH	4.64	8.59	6.34	0.99	0.46

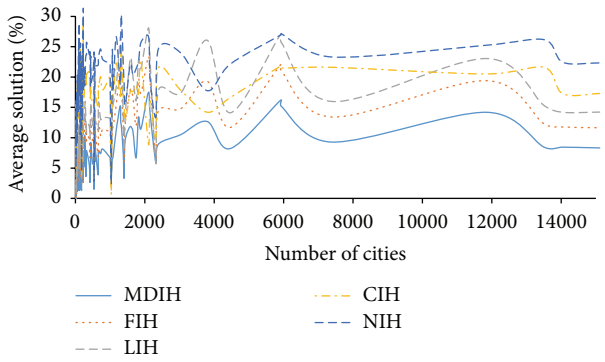

Figure 2: Graph showing average solution performance of heuristics.

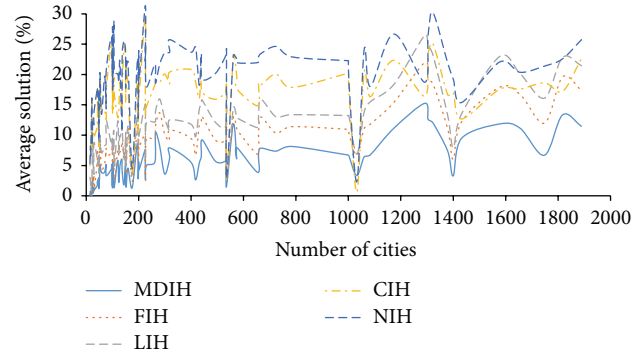

Figure 3: Graph showing average solution performance of heuristics up to 2000 city datasets.

While the CIH-NIH pattern is dominated by CIH. Notably, we can see that MDIH is not bettered by any of the other heuristics as we move beyond the 2000 city mark.

To make the picture clear for datasets up to the 2000 cities another graph in Figure 3 is presented. This graph removes the results of datasets more than 2000 cities to make picture clear for smaller problems. However even in this graph picture is not clear up to 200 cities as there is a dense accumulation of datasets up to this point. However picture beyond 200 mark is quiet clear. Again seemingly there are two patterns, one followed by CIH-NIH and another followed by MDIH-FIH-LIH. Former is dominated by CIH and latter is dominated by MDIH. In overall picture mostly MDIH graph dominates all the other graphs.

6.3. Experiments to Compare Performance of Cost Based Insertion Heuristics with Their Faster Counterparts. We turn now to the fast approximate variants of these heuristics. Table 4 shows comparative results for the standard and fast/approximate variants of CIH, LIH, and MDIH. From the table it can be seen that there are statistically only small differences between the standard and fast variants across all the four criteria. The largest difference between any standard and fast variant occurs in the case of CIH and is in favour of the faster method, with FCIH solution quality on average deviating 13.99% from optimal, compared with 14.26% for CIH. In all cases, the fast method naturally provides a significant advantage in execution time, with a reduction of 70–80% from the execution time of the standard variant.

We also performed statistical tests (Student's t-test) to investigate whether or not there was any significant difference in solution quality between the standard and faster versions. While comparing results between CIH and FCIH based on a confidence level of 95%, it was found that CIH

Table 3: Results of 5 different tour construction heuristics after removing noise (dataset BRG180).

Name	Average of best solution % age	Average of worst solution % age	Average of average solution % age	Average of standard deviation % age	Average of execution time seconds
(0)	(1)	(2)	(3)	(4)	(5)
NIH	17.78	22.37	20.10	1.25	0.63
CIH	14.26	18.82	16.61	1.25	0.48
LIH	8.88	15.97	12.01	1.74	0.42
FIH	6.84	12.89	9.51	1.52	0.64
MDIH	4.35	8.18	6.00	0.95	0.46

Table 4: Comparison of standard heuristics with their faster designs.

Name	Category	Average of best solution % age	Average of worst solution % age	Average of average solution % age	Average of standard deviation % age	Average of execution time seconds
(0a)	(0b)	(1)	(2)	(3)	(4)	(5)
CIH	Standard	14.26	18.82	16.61	1.25	0.48
	Fast	13.99	18.84	16.46	1.29	0.1
LIH	Standard	8.88	15.97	12.01	1.74	0.42
	Fast	9.02	16.19	12.24	1.79	0.11
MDIH	Standard	4.35	8.18	6.00	0.95	0.46
	Fast	4.3	8.27	6.02	0.99	0.14

performed better than FCIH in 17 out of 108 datasets, while FCIH performed better than CIH on 10 out of 108 datasets. Similarly, while comparing results between LIH and FLIH it was found that LIH performed better than FLIH on only 2 out of 108 datasets while FLIH performed better than LIH in 18 out of 108 datasets. While comparing results between MDIH and FMDIH it was found that MDIH performed better than FMDIH in only one out of 108 datasets while FMDIH performed better than MDIH in 3 out of 108 datasets. From these results it is clear that the new heuristics detailed in this paper are competitive in solution quality with the original slower versions; moreover, when there is a statistical difference in quality, this is more often than not (with the exception of CIH versus FCIH) in favour of the faster heuristic. We would conclude that the faster heuristics are generally worth applying to save computational time, with the risk to jeopardise solution quality being balanced by a similar chance of improved solution quality.

We note that a detailed performance comparison among the faster designs is not necessary, since the performance characteristics of the faster designs clearly echo those of the standard designs; therefore comparison among faster versions of heuristics reflects the same comparative analysis that was done among standard versions in Table 3.

6.4. Experiments to Compare Performance of FMDIH and FIH. Since MDIH has turned out to be best among all heuristics in the above experiments and its faster version FMDIH has provided same solution quality equal to MDIH with reduced time complexity equal to FIH, this shows a leap forward in the achievement of solution quality with time complexity of $O(n^2)$. To test whether or not this advantage is statistically significant we have given comparison of FMDIH with FIH in Appendix with full detail and commentary along with statistical tests. In terms of the columns in Tables 2–4, we simply note here that the FMDIH summary values are, in order of the columns, as follows: 4.30, 8.27, 6.02, 0.99, and 0.14. These are to be compared with the FIH summary values from Table 3, which are, respectively, 6.84, 12.89, 9.51, 1.52, and 0.64. Here however we visualise the relative performance of FMDIH and FIH (previously the best reported $O(n^2)$ tour construction heuristic) with Figure 4, again using the same axes as in Figures 2-3. It can be seen in Figure 4 that FMDIH is consistently better than FIH throughout the dataset spectrum, with the advantage even apparent in the dense area of problem instances smaller than 2000 cities.

7. Conclusions and Future Work

In this paper a previously unpublished tour construction heuristic, namely, the largest insertion heuristic (LIH), is introduced. It is shown that LIH completes the set of well-known popular tour insertion heuristics (cheapest, nearest, and farthest) and seems to produce better solution quality than cheapest and nearest insertion heuristics on the great majority of datasets. Furthermore, this paper also introduces

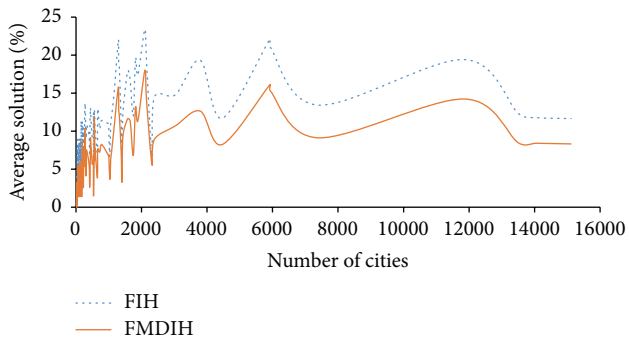

Figure 4: Graph showing performance comparison between FMDIH and FIH.

max difference insertion heuristic (MDIH). It is based on a max difference concept previously introduced in [11] as a way to engineer variants of other heuristics; however in this paper we have argued that, following its "completion" by defining an initial subtour establishment method, it merits "first-class" status as a heuristic in itself, alongside others of the same type. The resulting heuristic MDIH produced better solutions on average than all the other baseline heuristics considered in this paper. Furthermore, we have introduced complexity curtailing techniques that reduced worst case complexity of cost based insertion heuristics, that is, CIH, LIH, and MDIH, from $O(n^3)$ to $O(n^2)$, resulting in the development of their faster versions FCIH, FLIH, and FMDIH. It is shown that the faster versions do not statistically produce inferior solutions but reduce computational cost up to 80%. The development of FMDIH proved a leap forward in production of quality solutions among community of $O(n^2)$ complexity heuristics. This is because it has performed 3.5% on average better than FIH on a wide spectrum of popular datasets. We believe this is the first time that FIH has been demonstrated to be outperformed by an $O(n^2)$ heuristic on a wide spectrum of popular test bed. Our immediate future work in this direction is inspired by complexity curtailing techniques. What if these techniques are introduced in other higher order heuristics to reduce their complexity. We believe more interesting results are yet to come.

Appendix

Comparison between FIH and FMDIH

This appendix presents detailed dataset-wise comparison between two tour construction heuristics, that is, farthest insertion heuristic (FIH) and fast max difference insertion heuristic (FMDIH). Both heuristics have complexity of the order $O(n^2)$. However FMDIH designed in this paper has shown consistently better performance in the wide spectrum of datasets in the popular test bed. In Table 5 column-2 shows the results of FIH and column-3 shows the results of FMDIH. Column (a) shows best value among 30 simulations and its percentage difference from the optimal value. Column (b) shows worst value among 30 simulations and its percentage difference from the optimal value. Column (c) shows average value of 30 simulations and its percentage difference from the optimal value. Column (d) shows standard deviation of 30 simulations and its value as a percentage of the optimal value. Column (e) shows average execution time of 30 simulations. The datasets are arranged according to size from smallest (14 cities) to largest (15112 cities). From the column (a) of the table it can be seen that FMDIH has obtained optimal solution in 9 datasets against 7 datasets by FIH. FMDIH has obtained better solution than FIH in 94 out of 108 datasets and in 6 datasets they have solution of same quality. This means that FMDIH has worse solution than FIH in only 8 datasets out of 108 datasets. From the column (b) of the table it can be seen that FMDIH has worse solutions of better quality than FIH in 105 out of 108 datasets. In two datasets they have worse solutions of equal quality. It means only in one dataset FIH has worse solution better than FMDIH. From column (c) of the table it can be seen that average solution of FMDIH is better in 105 out of 108 datasets. This means FMDIH is beaten only in 3 datasets in terms of average solution. From the column (d) it can be seen that FMDIH has better standard deviation than FIH in 96 out of 108 datasets. This means only on 12 datasets FMDIH is worse in standard deviation than FIH. The story does not stop here; there are more interesting comparisons. By comparing column 3C with column 2a it can be seen that FMDIH has obtained average solutions in 69 out of 108 datasets that are better than the best solutions of FIH. However FIH has not attained this feet on any of the datasets. By comparing column 3b with 2c it can be seen that worst solutions of FMDIH are better than the average solution of FIH in 79 out of 108 datasets, while in none of the datasets FIH has attained this feet. Now most interesting result can be seen by comparing column 3b with column 2a. The worst solutions obtained by FMDIH in 30 simulations are better than the best solutions obtained by FIH in 30 simulations in 28 out of 108 datasets (>25%). If results are looked at more carefully then it can be seen that these 28 datasets start from RD400 dataset. There are only 49 datasets that are of the size 400 cities and above. In 28 out of these 49 datasets (>50%) worst solutions of FMDIH are better than the best solutions of FIH. If dataset spectrum is squeezed to larger band of 1000 cities and above then it is very surprising to see FMDIH has attained worse solutions better than the best solutions of FIH in 25 out of 31 datasets (>80%). Finally in all the 8 datasets of greater than 4000 cities worst solutions of FMDIH are better than the best solutions of FIH in 30 simulations. These results show that advantage to FMDIH improves with increase in size of the dataset and this clearly put performance of FMDIH much ahead of FIH. Student's t-test was conducted which confirmed that the output of FMDIH is better than FIH up to the 99.5% of confidence level on the 100 datasets. On another dataset FMDIH had better results up to 95% of confidence level, where FIH did not prove any statistical supremacy on any of the 108 datasets.

Competing Interests

The authors declare that they have no competing interests.

Table 5

Dataset	Optimal	Farthest insertion heuristic									Fast max difference insertion heuristic								
(1)		(2)									(3)								
		Min	% diff.	Max	% diff.	Avg.	% diff.	Stdv	% age	Time	Min	% diff.	Max	% diff.	Avg.	% diff.	Stdv	% age	Time
(a)	(b)	(a)		(b)		(c)		(d)		(e)	(a)		(b)		(c)		(d)		(e)
Burma14	3323	3323	0.00	3442	3.58	3327.83	0.15	21.45	0.65	0.00	3323	0.00	3442	3.58	3326.97	0.12	21.36	0.64	0.00
Ulysses16	6859	6859	0.00	7082	3.25	6900.67	0.61	59.97	0.87	0.00	6859	0.00	7082	3.25	6912.50	0.78	58.89	0.86	0.00
Gr17	2085	2095	0.48	2162	3.69	2102.17	0.82	19.96	0.96	0.00	2085	0.00	2159	3.55	2095.73	0.51	21.58	1.04	0.00
GR21	2707	2707	0.00	2816	4.03	2754.80	1.77	48.57	1.79	0.00	2707	0.00	2707	0.00	2707.00	0.00	0.00	0.00	0.00
Ulysses22	7013	7013	0.00	7592	8.26	7116.03	1.47	131.65	1.88	0.00	7013	0.00	7262	3.55	7088.67	1.08	62.04	0.88	0.00
GR24	1272	1272	0.00	1387	9.04	1322.97	4.01	26.50	2.08	0.00	1272	0.00	1341	5.42	1313.93	3.30	15.82	1.24	0.00
Fri26	937	937	0.00	1021	8.96	957.77	2.22	20.49	2.19	0.00	937	0.00	958	2.24	941.40	0.47	8.00	0.85	0.00
Bayg29	1610	1628	1.12	1746	8.45	1664.53	3.39	28.17	1.75	0.00	1618	0.50	1668	3.60	1631.97	1.36	10.48	0.65	0.00
Bays29	2020	2028	0.40	2083	3.12	2035.80	0.78	12.44	0.62	0.00	2033	0.64	2068	2.38	2038.40	0.91	6.08	0.30	0.00
Dantzig42	699	712	1.86	761	8.87	738.83	5.70	13.07	1.87	0.00	705	0.86	743	6.29	711.63	1.81	8.70	1.25	0.00
Swiss42	1273	1319	3.61	1439	13.04	1371.40	7.73	31.35	2.46	0.00	1273	0.00	1346	5.73	1302.67	2.33	25.60	2.01	0.00
ATT48	10628	10845	2.04	11554	8.71	11205.10	5.43	193.80	1.82	0.00	10653	0.24	11073	4.19	10929.70	2.84	123.93	1.17	0.00
GR48	5046	5099	1.05	5441	7.83	5271.83	4.48	114.17	2.26	0.00	5109	1.25	5295	4.93	5195.73	2.97	61.40	1.22	0.00
HK48	11461	11474	0.11	12513	9.18	11998.70	4.69	316.11	2.76	0.00	11556	0.83	11982	4.55	11611.10	1.31	94.18	0.82	0.00
Eil51	426	437	2.58	462	8.45	450.77	5.81	6.99	1.64	0.00	434	1.88	443	3.99	439.67	3.21	2.33	0.55	0.00
Berlin52	7542	7542	0.00	8677	15.05	8214.50	8.92	316.31	4.19	0.00	7604	0.82	8425	11.71	7948.97	5.40	239.48	3.18	0.00
Brazil58	25395	25664	1.06	27452	8.10	26529.20	4.47	378.63	1.49	0.00	25717	1.27	26546	4.53	26298.90	3.56	196.66	0.77	0.00
ST70	675	696	3.11	736	9.04	710.30	5.23	9.85	1.46	0.00	682	1.04	717	6.22	700.30	3.75	9.18	1.36	0.00
Eil76	538	570	5.95	598	11.15	580.40	7.88	9.64	1.79	0.00	556	3.35	577	7.25	566.20	5.24	5.40	1.00	0.00
PR76	108159	110516	2.18	126235	16.71	115154.00	6.47	4255.54	3.93	0.00	109796	1.51	115392	6.69	111760.00	3.33	1756.05	1.62	0.00
GR96	55209	56854	2.98	63009	14.13	59177.70	7.19	1419.23	2.57	0.00	55738	0.96	59941	8.57	57658.00	4.44	801.67	1.45	0.00
Rat99	1211	1257	3.80	1382	14.12	1313.20	8.44	32.08	2.65	0.00	1244	2.73	1296	7.02	1272.03	5.04	12.63	1.04	0.00
KroA100	21282	22026	3.50	23487	10.36	22608.10	6.23	392.16	1.84	0.00	21478	0.92	22599	6.19	21628.70	1.63	217.73	1.02	0.00
KroB100	22141	22693	2.49	24518	10.74	23508.10	6.17	396.25	1.79	0.00	22211	0.32	23405	5.71	22949.90	3.65	282.76	1.28	0.00
KroC100	20749	21105	1.72	22599	8.92	21853.70	5.32	375.72	1.81	0.00	20749	0.00	21838	5.25	21045.70	1.43	225.82	1.09	0.00
KroD100	21294	21745	2.12	23374	9.77	22528.60	5.80	376.69	1.77	0.00	21403	0.51	22708	6.64	21880.20	2.75	286.35	1.34	0.00
KroE100	22068	22514	2.02	24147	9.42	23229.20	5.26	484.60	2.20	0.00	22179	0.50	23195	5.11	22749.20	3.09	300.23	1.36	0.00
RD100	7910	8255	4.36	8936	12.97	8619.23	8.97	167.00	2.11	0.00	7965	0.70	8380	5.94	8158.47	3.14	111.72	1.41	0.00
Eil101	629	660	4.93	701	11.45	679.60	8.04	10.61	1.69	0.00	648	3.02	683	8.59	664.60	5.66	7.90	1.26	0.00
Lin105	14379	14863	3.37	16088	11.89	15435.30	7.35	309.15	2.15	0.00	14402	0.16	15179	5.56	14708.30	2.29	182.60	1.27	0.00
PR107	44303	45030	1.64	45933	3.68	45450.50	2.59	245.78	0.55	0.00	44489	0.42	45420	2.52	44925.80	1.41	216.48	0.49	0.00
GR120	6942	7162	3.17	7817	12.60	7462.97	7.50	162.80	2.35	0.00	7164	3.20	7494	7.95	7323.47	5.50	79.69	1.15	0.00
PR124	59030	60305	2.16	65845	11.54	62875.70	6.51	1339.81	2.27	0.00	59767	1.25	62202	5.37	60323.20	2.19	625.42	1.06	0.00
Bier127	118282	123796	4.66	130733	10.53	126795.00	7.20	1891.57	1.60	0.00	121130	2.41	131232	10.95	123455.00	4.37	1996.69	1.69	0.00
CH130	6110	6335	3.68	6884	12.67	6556.80	7.31	139.36	2.28	0.00	6328	3.57	6686	9.43	6441.50	5.43	78.33	1.28	0.00
PR136	96772	102573	5.99	107287	10.87	105143.00	8.65	1129.12	1.17	0.00	99960	3.29	104166	7.64	102520.00	5.94	975.59	1.01	0.00
GR137	69853	73391	5.06	81011	15.97	76135.90	8.99	1606.32	2.30	0.00	71773	2.75	74864	7.17	72760.50	4.16	785.68	1.12	0.00
PR144	58537	59336	1.36	64705	10.54	62041.30	5.99	1383.72	2.36	0.00	59005	0.80	63056	7.72	59680.10	1.95	807.68	1.38	0.00
CH150	6528	6864	5.15	7479	14.57	7094.67	8.68	148.20	2.27	0.00	6733	3.14	7103	8.81	6901.07	5.71	97.47	1.49	0.00

Table 5: Continued.

Dataset	Optimal	Farthest insertion heuristic									Fast max difference insertion heuristic								
	(1)	(2)									(3)								
		Min	% diff.	Max	% diff.	Avg.	% diff.	Stdv	% age	Time	Min	% diff.	Max	% diff.	Avg.	% diff.	Stdv	% age	Time
(a)	(b)	(a)		(b)		(c)		(d)		(e)	(a)		(b)		(c)		(d)		(e)
kroA150	26524	27407	3.33	29394	10.82	28514.90	7.51	544.26	2.05	0.00	27066	2.04	28285	6.64	27593.20	4.03	284.44	1.07	0.00
kroB150	26130	27337	4.62	28912	10.65	27859.30	6.62	375.99	1.44	0.00	26322	0.73	27868	6.65	27049.60	3.52	392.87	1.50	0.00
PR152	73682	74827	1.55	77967	5.82	76401.20	3.69	808.26	1.10	0.00	74029	0.47	76257	3.49	74715.40	1.40	615.89	0.84	0.00
U159	42080	44506	5.77	49335	17.24	46856.30	11.35	1195.20	2.84	0.00	43367	3.06	45621	8.41	44070.50	4.73	527.23	1.25	0.00
SI175	21407	21751	1.61	22193	3.67	21911.90	2.36	119.10	0.56	0.00	21527	0.56	21883	2.22	21707.30	1.40	112.23	0.52	0.00
Rat195	2323	2503	7.75	2686	15.63	2585.10	11.28	42.01	1.81	0.00	2487	7.06	2588	11.41	2525.47	8.72	25.35	1.09	0.00
d198	15780	16277	3.15	17207	9.04	16696.60	5.81	311.97	1.98	0.00	16015	1.49	16621	5.33	16214.40	2.75	161.35	1.02	0.00
KroA200	29368	30533	3.97	32481	10.60	31225.90	6.33	448.47	1.53	0.00	30258	3.03	31662	7.81	31002.70	5.57	372.36	1.27	0.00
KroB200	29437	31227	6.08	33304	13.14	32006.80	8.73	427.13	1.45	0.00	30387	3.23	31877	8.29	30987.90	5.27	438.01	1.49	0.00
GR202	40160	42038	4.68	45263	12.71	43423.10	8.13	725.72	1.81	0.00	40889	1.82	43698	8.81	41942.20	4.44	646.74	1.61	0.00
TS225	126643	134365	6.10	143384	13.22	139859.00	10.44	2128.75	1.68	0.00	135470	6.97	138995	9.75	136429.00	7.73	1115.29	0.88	0.00
TSP225	3916	4182	6.79	4501	14.94	4292.40	9.61	78.76	2.01	0.00	4118	5.16	4292	9.60	4187.50	6.93	45.32	1.16	0.00
PR226	80369	81020	0.81	84593	5.26	82441.70	2.58	848.57	1.06	0.00	81598	1.53	83216	3.54	82492.30	2.64	296.77	0.37	0.00
GR229	134602	142714	6.03	154567	14.83	147270.00	9.41	3066.84	2.28	0.00	138704	3.05	145318	7.96	141429.00	5.07	1740.97	1.29	0.00
Gil262	2378	2507	5.42	2683	12.83	2597.10	9.21	42.29	1.78	0.00	2448	2.94	2548	7.15	2514.13	5.72	24.39	1.03	0.00
PR264	49135	53814	9.52	55504	12.96	54821.70	11.57	425.79	0.87	0.00	53021	7.91	55170	12.28	54211.90	10.33	508.62	1.04	0.00
A280	2579	2847	10.39	3043	17.99	2931.93	13.68	50.88	1.97	0.00	2691	4.34	2861	10.93	2792.60	8.28	44.26	1.72	0.00
PR299	48191	51625	7.13	56160	16.54	53082.40	10.15	985.20	2.04	0.00	48763	1.19	52054	8.02	50174.00	4.11	747.97	1.55	0.00
Lin318	42029	44730	6.43	46981	11.78	45840.00	9.07	575.64	1.37	0.00	43516	3.54	45306	7.80	44471.20	5.81	506.01	1.20	0.00
Linhp318	41345	44730	8.19	46981	13.63	45840.00	10.87	575.64	1.39	0.00	43516	5.25	45306	9.58	44471.20	7.56	506.01	1.22	0.00
RD400	15281	16441	7.59	17139	12.16	16724.50	9.45	164.65	1.08	0.00	15876	3.89	16431	7.53	16132.40	5.57	128.14	0.84	0.00
FL417	11861	12201	2.87	13147	10.84	12657.10	6.71	245.19	2.07	0.00	12061	1.69	12214	2.98	12171.80	2.62	32.24	0.27	0.00
GR431	171414	181584	5.93	200082	16.72	191434.00	11.68	5458.44	3.18	0.00	177362	3.47	183560	7.09	180437.00	5.26	1412.80	0.82	0.00
PR439	107217	116581	8.73	124415	16.04	120425.00	12.32	2176.87	2.03	0.00	110900	3.44	117753	9.83	114517.00	6.81	1734.12	1.62	0.00
PCB442	50778	55660	9.61	59032	16.26	57419.30	13.08	736.18	1.45	0.00	54452	7.24	56840	11.94	55421.30	9.14	550.44	1.08	0.00
D493	35002	37545	7.27	39044	11.55	38212.00	9.17	440.14	1.26	0.00	36402	4.00	37732	7.80	36984.00	5.66	294.40	0.84	0.00
ATT532	27686	29670	7.17	31101	12.33	30257.50	9.29	360.98	1.30	0.00	28797	4.01	29796	7.62	29293.20	5.81	242.24	0.87	0.00
Ali535	202339	216002	6.75	238252	17.75	227611.00	12.49	6102.76	3.02	0.01	210768	4.17	221967	9.70	216135.00	6.82	2858.38	1.41	0.01
SI535	48450	49496	2.16	50025	3.25	49816.20	2.82	148.87	0.31	0.00	49020	1.18	49346	1.85	49167.90	1.48	83.64	0.17	0.00
PA561	2763	3061	10.79	3204	15.96	3118.40	12.86	29.61	1.07	0.01	3007	8.83	3141	13.68	3092.17	11.91	33.81	1.22	0.01
U574	36905	40097	8.65	42081	14.03	40969.00	11.01	370.18	1.00	0.01	38752	5.00	39928	8.19	39252.00	6.36	346.62	0.94	0.01
Rat575	6773	7374	8.87	7594	12.12	7508.17	10.85	53.59	0.79	0.01	7191	6.17	7444	9.91	7300.83	7.79	58.16	0.86	0.01
P654	34643	35659	2.93	41131	18.73	37016.60	6.85	1376.23	3.97	0.01	35334	1.99	36176	4.43	35971.10	3.83	207.24	0.60	0.01
D657	48912	52482	7.30	55079	12.61	53826.00	10.05	650.07	1.33	0.01	51459	5.21	53624	9.63	52371.20	7.07	430.21	0.88	0.01
GR666	294358	324349	10.19	341937	16.16	332601.00	12.99	4859.02	1.65	0.01	311264	5.74	328227	11.51	317068.00	7.72	3965.46	1.35	0.01
U724	41910	45425	8.39	47487	13.31	46237.80	10.33	470.09	1.12	0.01	44499	6.18	45481	8.52	44946.70	7.25	277.31	0.66	0.01
Rat783	8806	9607	9.10	9959	13.09	9809.83	11.40	70.02	0.80	0.01	9419	6.96	9665	9.75	9532.33	8.25	62.85	0.71	0.01
DSJ1000	18659688	$2.04E+07$	9.33	$2.12E+07$	13.46	20705300.00	10.96	199033.00	1.07	0.03	$1.97E+07$	5.63	$2.02E+07$	8.51	19916900.00	6.74	138698.00	0.74	0.02
DSJ1000_Ceil	18660188	$2.04E+07$	9.33	$2.12E+07$	13.46	20705900.00	10.96	199248.00	1.07	0.03	$1.97E+07$	5.63	$2.02E+07$	8.51	19914600.00	6.72	136854.00	0.73	0.02

Table 5: Continued.

Dataset	Optimal	Farthest insertion heuristic									Fast max difference insertion heuristic								
	(1)	(2)									(3)								
		Min	% diff.	Max	% diff.	Avg.	% diff.	Stdv	% age	Time	Min	% diff.	Max	% diff.	Avg.	% diff.	Stdv	% age	Time
(a)	(b)	(a)		(b)		(c)		(d)		(e)	(a)		(b)		(c)		(d)		(e)
Pr1002	259045	279938	8.07	290550	12.16	285664.00	10.28	2895.93	1.12	0.03	270964	4.60	280759	8.38	276781.00	6.85	2215.42	0.86	0.02
Si1032	92650	98444	6.25	99088	6.95	98819.50	6.66	154.80	0.17	0.03	95552	3.13	96430	4.08	96050.60	3.67	196.73	0.21	0.02
U1060	224094	245679	9.63	254238	13.45	248951.00	11.09	2035.57	0.91	0.03	235078	4.90	241873	7.93	238367.00	6.37	1861.23	0.83	0.02
Vm1084	239297	262043	9.51	274616	14.76	267609.00	11.83	2850.55	1.19	0.03	253819	6.07	262898	9.86	256516.00	7.20	1871.48	0.78	0.02
Pcb1173	56892	64918	14.11	66795	17.41	65640.20	15.38	442.92	0.78	0.04	62530	9.91	64149	12.76	63103.30	10.92	358.86	0.63	0.02
D1291	50801	60221	18.54	63234	24.47	61956.00	21.96	614.26	1.21	0.05	57501	13.19	59945	18.00	58810.80	15.77	641.59	1.26	0.03
RL1304	252948	292602	15.68	308095	21.80	301514.00	19.20	3688.25	1.46	0.05	279047	10.32	291228	15.13	285207.00	12.75	3363.99	1.33	0.03
RL1323	270199	312400	15.62	328087	21.42	320036.00	18.44	4117.85	1.52	0.05	297661	10.16	307528	13.82	302587.00	11.99	2445.58	0.91	0.03
NRW1379	56638	62208	9.83	63899	12.82	62863.50	10.99	420.18	0.74	0.06	60170	6.24	61393	8.40	60785.70	7.32	279.13	0.49	0.03
FL1400	20127	20949	4.08	21909	8.85	21338.30	6.02	273.56	1.36	0.06	20516	1.93	21201	5.34	20786.70	3.28	213.61	1.06	0.03
U1432	152970	168810	10.35	174677	14.19	171655.00	12.21	1166.59	0.76	0.06	165373	8.11	169884	11.06	167195.00	9.30	1092.88	0.71	0.03
FL1577	22249	25155	13.06	27824	25.06	26228.30	17.89	514.47	2.31	0.07	24030	8.00	25892	16.37	24835.70	11.63	445.83	2.00	0.04
D1655	62128	71507	15.10	73675	18.59	72508.90	16.71	530.78	0.85	0.08	68217	9.80	70258	13.09	69165.90	11.33	447.10	0.72	0.04
VM1748	336556	371239	10.31	381960	13.49	376862.00	11.98	3094.47	0.92	0.10	355083	5.50	364325	8.25	359501.00	6.82	2503.97	0.74	0.05
U1817	57201	67612	18.20	69628	21.73	68406.70	19.59	596.43	1.04	0.10	63383	10.81	65642	14.76	64709.40	13.13	519.68	0.91	0.05
RL1889	316536	365077	15.34	376387	18.91	371580.00	17.39	3230.94	1.02	0.12	348074	9.96	359023	13.42	352505.00	11.36	2975.08	0.94	0.06
D2103	80450	98109	21.95	100527	24.96	99264.90	23.39	657.60	0.82	0.14	93544	16.28	95910	19.22	94908.90	17.97	527.46	0.66	0.07
U2152	64253	75721	17.85	77597	20.77	76732.50	19.42	527.56	0.82	0.15	72492	12.82	74131	15.37	73311.70	14.10	430.31	0.67	0.07
U2319	234256	247881	5.82	251176	7.22	250028.00	6.73	715.57	0.31	0.17	245939	4.99	248584	6.12	247465.00	5.64	687.33	0.29	0.09
Pr2392	378032	428627	13.38	438782	16.07	432695.00	14.46	2243.77	0.59	0.19	406210	7.45	417522	10.45	411749.00	8.92	2681.97	0.71	0.09
Pcb3038	137694	156708	13.81	159518	15.85	158069.00	14.80	586.58	0.43	0.32	151346	9.91	153766	11.67	152435.00	10.71	603.48	0.44	0.15
FL3795	28772	33124	15.13	35354	22.88	34331.30	19.32	542.79	1.89	0.49	31317	8.85	33137	15.17	32407.00	12.63	585.13	2.03	0.24
FNL4461	182566	202328	10.82	204937	12.25	203925.00	11.70	601.55	0.33	0.74	196918	7.86	198936	8.97	197617.00	8.24	495.01	0.27	0.32
RL5915	565530	683054	20.78	697158	23.28	690217.00	22.05	3493.67	0.62	1.39	648152	14.61	662852	17.21	656416.00	16.07	3429.91	0.61	0.56
RL5934	556045	660399	18.77	681665	22.59	672930.00	21.02	4608.88	0.83	1.40	630512	13.39	649307	16.77	641010.00	15.28	3805.45	0.68	0.56
pla7397	23260728	26066700	12.06	26640000	14.53	26382900.00	13.42	143431.00	0.62	2.32	25178800	8.25	25559100	9.88	25383600.00	9.13	87578.50	0.38	0.86
rl11849	923288	1093120	18.39	1108780	20.09	1102520.00	19.41	3886.70	0.42	10.11	1046110	13.30	1060200	14.83	1054670.00	14.23	3259.40	0.35	2.23
usa13509	19982859	22318200	11.69	22561500	12.90	22442600.00	12.31	59189.90	0.30	15.01	21566100	7.92	21768200	8.93	21673900.00	8.46	50429.90	0.25	2.88
brd14051	469385	522027	11.22	527320	12.34	524714.00	11.79	1160.75	0.25	16.43	506838	7.98	510569	8.77	508872.00	8.41	910.74	0.19	3.12
d15112	1573084	1750830	11.30	1762650	12.05	1756490.00	11.66	2965.69	0.19	19.11	1698080	7.95	1707830	8.57	1704060.00	8.33	2482.61	0.16	3.61
			6.84		**12.89**		**9.51**		**1.52**	**0.64**		**4.30**		**8.27**		**6.02**		**0.99**	**0.14**

Acknowledgments

The authors are grateful for financial support from Innovate UK and Route Monkey Ltd. via KTP partnership no. 9839.

References

[1] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The Traveling Salesman Problem*, Princeton University Press, 2007.

[2] J. K. Lenstra and A. H. G. R. Kan, "Some simple applications of the travelling salesman problem," *Operational Research Quarterly*, vol. 26, no. 4, pp. 717–733, 1975.

[3] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations Research*, vol. 21, no. 2, pp. 498–516, 1973.

[4] http://www.math.uwaterloo.ca/tsp/concorde/index.html.

[5] J. Perttunen, "On the significance of the initial solution in travelling salesman heuristics," *The Journal of the Operational Research Society*, vol. 45, no. 10, pp. 1131–1140, 1994.

[6] D. J. Rosenkrantz, R. E. Stearns, and I. Lewis, "An analysis of several heuristics for the traveling salesman problem," *SIAM Journal on Computing*, vol. 6, no. 3, pp. 563–581, 1977.

[7] D. S. Johnson and L. A. McGeoch, "Experimental analysis of heuristics for the STSP," in *The Traveling Salesman Problem and Its Variations*, G. Gutin and A. P. Punnen, Eds., vol. 12 of *Combinatorial Optimization*, chapter 9, pp. 369–443, Kulwer Academic Publishers, 2002.

[8] J. J. Bently, "Fast algorithms for the geometric travelling salesman problem," *ORSA Journal on Computing*, vol. 4, no. 4, pp. 387–411, 1992.

[9] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: a case study in local optimization," in *Local Search in Combinatorial Optimization*, vol. 1, pp. 215–310, John Wiley & Sons, 1997.

[10] T. A. J. Nicholson, "A sequential method for discrete optimization problems and its application to the assignment, travelling salesman, and three machine scheduling problems," *IMA Journal of Applied Mathematics*, vol. 3, no. 4, pp. 362–375, 1967.

[11] G. Righini, *The Largest Insertion algorithm for the Traveling Salesman Problem*, 2000, http://crema.di.unimi.it/~righini/Didattica/Algoritmi%20Euristici/MaterialeAE/larg_ins.pdf.

[12] T. W. Tunnel and L. Heath, *Development of new heuristics for the Euclidean travelling salesman problem [M.S. thesis]*, Virginia Polytechnic Institute and State University, Blacksburg, Va, USA, 1989, http://eprints.cs.vt.edu/archive/00000167/01/TR-89-30.pdf.

[13] G. Reinelt, *The Traveling Salesman Problem: Computational Solutions for TSP Applications*, vol. 840 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 1994.

[14] B. L. Golden and W. Stewart Jr., "Empirical analysis of heuristics," in *The Travelling Salesman Problem*, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnoy Kan, and D. B. Shimoys, Eds., John Wiley & Sons, London, UK, 1985.